1996

# Automatic Generation of 3D CAD Models

Chandrajit L. Bajaj

Fausto Bernardini

Jindon Chen

Daniel R. Schikore

Report Number:
96-015

# AUTOMATIC GENERATION OF 3D CAD MODELS

**Chandrajit L. Bajaj**
**Fausto Bernardini**
**Jindon Chen**
**Daniel R. Schikore**

**Department of Computer Sciences**
**Purdue University**
**West Lafayette, IN 47907**

**CSD TR-96-015**
**February 1996**

# Automatic Generation of 3D CAD Models *

Chandrajit L. Bajaj Fausto Bernardini
Jindon Chen Daniel R. Schikore

Department of Computer Sciences
Purdue University

West Lafayette, IN 47907-1398 USA.
Tel: (317) 494-6531
Email: {bajaj,fxb,jdc,drs}@cs.purdue.edu

## Abstract

We present an approach for the reconstruction and approximation of 3D CAD models from an unorganized collection of points. Applications include rapid reverse engineering of existing objects for use in a synthetic computer environment, including computer aided design and manufacturing. Our reconstruction approach is flexible enough to permit interpolation of both smooth surfaces and sharp features, while placing few restrictions on the geometry or topology of the object.

Our algorithm is based on 3D Delaunay triangulations and $\alpha$-shapes to compute an initial triangle mesh approximating the shape of the object. A mesh decimation technique is applied to the dense triangle mesh to build a simplified approximation, while retaining important topological and geometric characteristics of the model. The reduced mesh is interpolated with piecewise algebraic surface patches, which approximate the original points.

The resulting model is suitable for typical CAD modeling and analysis applications.

## 1 Introduction

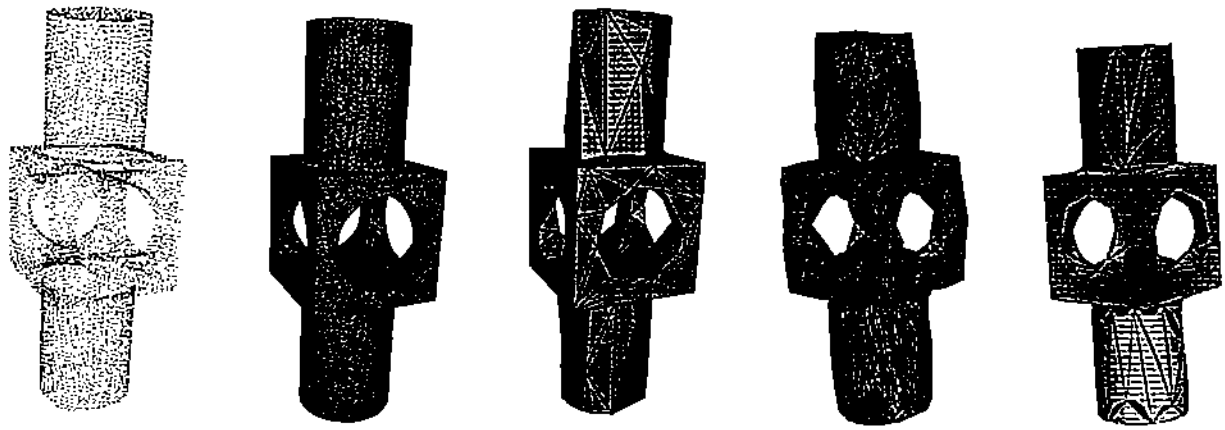
The design, engineering and manufacturing planning of new products is more and more often carried out by computer simulation. A common need in this process is incorporating existing objects into this *electronic prototyping* environment, to reuse them as part of a new product, or to adapt and improve their design to meet new requirements.

The availability of fast and accurate geometric data acquisition devices, such as the *laser range scanner*, has made it relatively simple to acquire the spatial coordinates of a large set of points from the surface of a 3D object. Reconstructing a CAD model of the object from this set of points is the subject of this paper. Among the qualities we would expect from such a model are the following:

- It matches the *topological* characteristics of the object
- It is geometrically accurate
- It can represent both smooth, curvature continuous surfaces and sharp features such as corners and edges, common in manufactured parts
- It is suitable to be used in successive phases of the design and simulation process

Our algorithm is based on 3D Delaunay triangulations and $\alpha$-shapes [12] for extracting a triangle mesh from the set of points. $\alpha$-shapes give a sound mathematical definition to the vague concept of *shape of a set of points*, and can be extended to *weighted* sets of points to allow adaptive sampling (i.e. using a large number of points, with small associated weight, in complicated areas, and a coarser sampling, with larger weight, in simpler regions). We then apply a mesh-decimation algorithm to reduce the number of triangles in the mesh. The decimation is carried out in a way that preserves sharp features and bounds the distance of the decimated mesh from the original set of points, while maintaining a good triangle aspect-ratio. The reduced mesh is used as the starting points for a polynomial-patch data fitting. For every triangle, we build an implicit Bernstein-Bézier patch of

*Supported in part by AFOSR grant F49620-94-1-0080, NSF grant CCR 92-22467, and ONR grant N00014-94-1-0370



(a) (b) (c) (d) (e)

FIGURE 1: Some of the steps required by our algorithm are depicted in this figure. The object has been sampled at about $10^4$ points (a). An $\alpha$-solid is computed (b) and then decimated (c). Finally, polynomial patches are used to fit the data. Figure (d) shows the simplicial hull construction (see Section 5.1. The final reconstructed model is visilble in (e).

low degree that interpolates the vertices and vertex normals (if defined) and least-squares approximates data points in its vicinity. The algebraic patches used (cubic A-patches [2, 3]) allow a simple formulation of $C^1$ continuity constraints between adjacent patches, and can model sharp features such as rectilinear sharp edges, piecewise-planar curvilinear creases and sharp corners.

The problem of reconstructing an approximation of an unknown manifold has been investigated by many authors (for a review, see [6]). Several works focus on building a piecewise-linear surface [7, 8, 15, 21, 33]. Other methods use parametric or functional surface patches for either local or global interpolation [5, 16, 20, 23, 26, 27]. The three-step method of Hoppe *et al.* [21, 22, 20] is based on reconstructing the shape of the object from an estimated *signed-distance*, followed by an optimization of the piecewise-linear mesh and by a data fitting step that produces a subdivision surface.

A few papers (see [1, 9, 10, 17, 25, 31]) use implicit surface patches.

Some of the advantages of our method with respect to existing techniques are the following: (i) Our algorithm does not require costly global optimizations, and is therefore quite suitable for practical use. (ii) The reconstructed model is in a form that can be easily used in successive analysis or modeling steps.

## 2 An Overview of the Reconstruction Algorithm

Our algorithm is based on the following three phases:

1. Build an initial triangle mesh that interpolates all data points, approximating the object shape. We explain in Section 3 how to extract a triangulated $\alpha$-solid from the data points (a technique based on $\alpha$-shapes [12]). The triangle mesh can be used to estimate normals at *smooth* vertices (by averaging the normals of incident triangles) and to detect sharp features (by looking at the dihedral angle formed by two adjacent triangles). Notice that for the dense surface sampling that we are interested in, these estimates are usually quite accurate.

2. Decimate the mesh to reduce the number of triangles, while guaranteeing good aspect-ratio of triangles, bounded distance of the data points from the reduced mesh, and feature preservation. The technique used in our paper has been adapted from [4]. The edges and vertices of the reduced mesh will be "tagged" as either *smooth* (the surface is $C^1$ continuous across it) or *sharp* (only $C^0$ continuity), and vertices are classified according to the type of incident edges and the number and type of estimated vertex normals.

Section 4 describes the mesh decimation algorithm.

3. Compute for each triangle in the mesh above a polynomial surface patch (an *A-patch*, see [3]). The patch interpolates the triangle vertices and estimated vertex normals. Additional degrees of freedom are used to least-square approximate other data points. We summarize definitions and properties of A-patches in Section 5.1, and detail this phase of the algorithm in Section 5.

An example of the reconstruction process is shown in Figure 1. The simple object shown will be used as a running example in the following Sections. (a) shows the original points. The $\alpha$-solid boundary and decimated triangle meshes are shown in (b) and (c). (d) and (e) show the final reconstruction (In (d) patches are randomly colored, and the *simplicial hull* (see Section 5.1) is shown).

# 3 Building an Initial Triangle Mesh

In the first pass of our algorithm we want to compute a triangle mesh, having the data points as vertices, that approximates the shape of the object. The approximation is based on the assumption that the point sampling is *dense* and *uniform*, so that one can exploit vicinity to infer spatial relationships among the unorganized sampled points. Our method is based on 3D $\alpha$-shapes [12]. An $\alpha$-complex of a set of points $P$, for a given value of the parameter $\alpha$, is a subset of the 3D Delaunay triangulation of $P$, and the corresponding $\alpha$-shape is its underlying space. Intuitively, an $\alpha$-shape can be obtained as follows: Consider a ball-shaped *eraser*, of radius $\sqrt{\alpha}$, and think of $P$ as a set of points in space that the eraser cannot inter-penetrate. The 3D Delaunay triangulation of $P$ is a simplicial complex formed by tetrahedra, triangles, edges and vertices (3-, 2-, 1- and 0-simplices, respectively). Imagine moving the eraser everywhere in space, removing all simplices that the eraser can pass through (remember that the eraser is constrained by the data points). All that is left after the erasing constitutes the $\alpha$-shape of $P$. Obviously, as $\alpha$ varies, one obtains different $\alpha$-shapes. For example, for $\alpha = 0$ the $\alpha$-shape is $P$ itself. For $\alpha = \infty$ one obtains the convex hull of $P$. In fact, Edelsbrunner *et al.* [12] show that varying $\alpha$ from 0 to $\infty$, a *finite* collection of $\alpha$-shapes is obtained. Notice that in general an $\alpha$-shape is a non-connected, non-regular (i.e., having solid as well as 2-, 1- and 0-dimensional parts) polytope, and therefore is not directly suitable for our purposes. We therefore define the concept of an $\alpha$-solid:

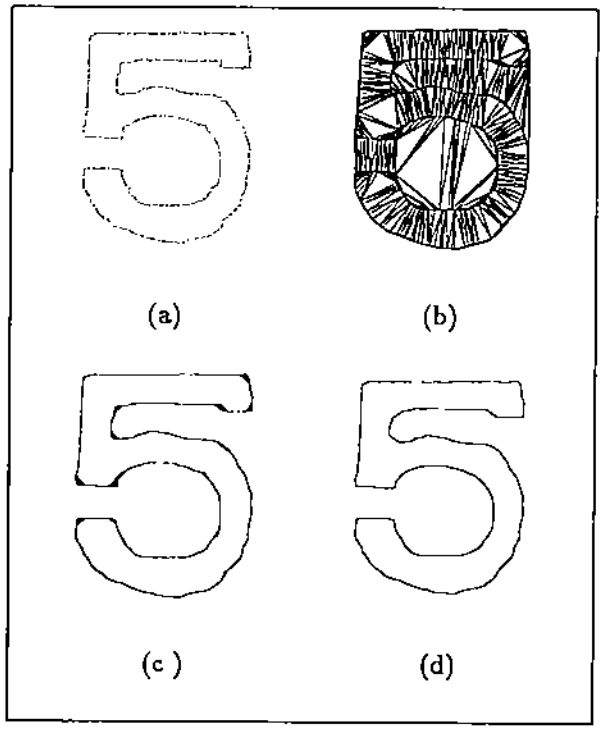



FIGURE 2: A simple example, for a 2D set of points (a), of the Delaunay triangulation (b), $\alpha$-shape (c) and $\alpha$-solid (d).

**Definition 3.1** *Given a point set $P$, its Delaunay triangulation $T$, and $\alpha$-complex $S_\alpha$, the* $\alpha$-solid *is the underlying space of the tetrahedra $\sigma \in T$ such that:*

1. *$\sigma \in S_\alpha$, or*
2. *Any path (a space curve) from a point $p$ in $\sigma$ to a point at "infinity" intersects a triangle $\tau \in S_\alpha$.*

Intuitively the $\alpha$-solid is the collection of tetrahedra that either belong to the $\alpha$-shape or are contained in a region bounded by a continuous collection of $\alpha$-shape triangles. Figure 2 shows a simple 2D example of an $\alpha$-shape and associated $\alpha$-solid.

**Choosing the right value of $\alpha$.** The $\alpha$-solid is what we are really interested in, as we need a closed, orientable manifold (the boundary of the polytope) as our initial mesh. However we need to select the value of $\alpha$ such that the $\alpha$-solid best matches the shape of the object. Depending on the value of $\alpha$, the boundary of the $\alpha$-solid can be empty, or formed

by several components, or not interpolate all data points. We apply the following heuristic, which has proved to work quite well in practice:

> *Select the smallest value of $\alpha$ for which the $\alpha$-solid is connected and contains all points.*

Sometimes, due to an insufficient sampling in areas where small features are present, some points might become internal to the selected $\alpha$-solid, and therefore will not be interpolated by the boundary triangles. The best solution for this problem would be to use *weighted* $\alpha$-shapes [11] and adaptive sampling. Alternatively, one could use some heuristics to locally modify the mesh and "snap" the triangle mesh to the internal points. We have observed however that for reasonably dense sampling this problem occurs rarely, and introduces very small errors, and therefore decided to simply ignore those internal points.

Figure 1(c) shows the $\alpha$-solid selected by the heuristic for our example object. More examples are shown in Section 6.

# 4 Decimating the Mesh

Our goal is to approximate the dense polygonal reconstructed surfaces with a small number of patches. We have found that the most efficient approach to this problem is a two-phase stepwise approach. We first compute a reduced polygonal mesh which represents the data to a desired degree of accuracy, and second compute a piecewise A-patch surface representation which interpolates the reduced mesh, approximates the original points, and meets with the desired continuity along the boundaries.

## 4.1 Mesh Reduction Overview

Surface mesh reduction is a general category of techniques designed to remove redundant information from a mesh. Geometric mesh reduction has been approached from several directions. Optimization methods [32, 22] have produced good results, but their time-intensive nature leads us to consider more practical methods. He *et al.* [19] sample a geometric object into a volume buffer which is low-pass filtered. Multiresolution surfaces are then extracted from the volume buffer using traditional isosurfacing techniques. A broad family of algorithms based on local point deletion and retriangulation based on geometric criteria are attractive for their ease of implementation, ability to capture sharp features, and fast performance [28, 18, 4].

In this work, we adapt the method of [4] to handle explicit edge feature detection and preservation. The resulting method is able to maintain a strict bound on the distance between the original mesh and the surface mesh, as well as maintaining sharp features in the reconstructed triangulation.

## 4.2 Algorithm Overview

The algorithm for reduction follows the basic strategy of other "vertex deletion" schemes, and can be summarized by the following steps:

1. initialize errors on all triangles to 0
2. for each vertex $v$
   (a) classify $v$ according to vertex configurations in figure 3
   (b) compute an initial triangulation of the neighbors of $v$
   (c) perform edge flipping to minimize the error in the new triangulation
   (d) compute sum of propagated error and introduced error for best new triangulation
   (e) if error is within user-specified bounds
       delete $v$ and surrounding triangles
       add triangles of new mesh
       update error values for introduced triangles

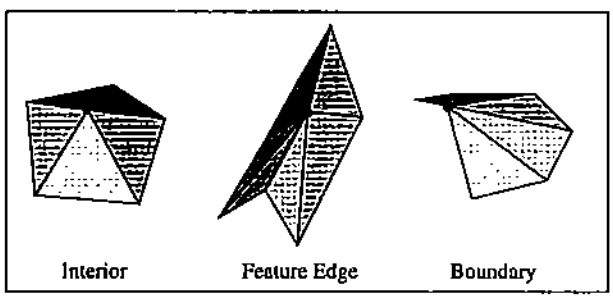



FIGURE 3: Types of candidate vertices

In the following subsections, we will briefly detail the steps of the algorithm and our extensions for sharp feature detection.

## 4.3 Mapping between triangulations

Errors introduced by deletion of a vertex $v$ are computed by establishing a mapping between the current triangulation about $v$ and the retriangulation of the neighbors of $v$. This is accomplished by normal projection of the new triangulation onto the original surface as indicated in figure 4. As shown in the figure, this projected triangulation effectively segments the region surrounding a vertex into simple wedge shapes. Due to the simple linear wedges, the maximal error (distance between triangulations) occurs at the intersections of new edges with the original edges, and need only be computed at these points. In one exceptional case, the triangle which contains the vertex $v$ which is being deleted, the error must also be computed between $v$ and the point which projects to it.

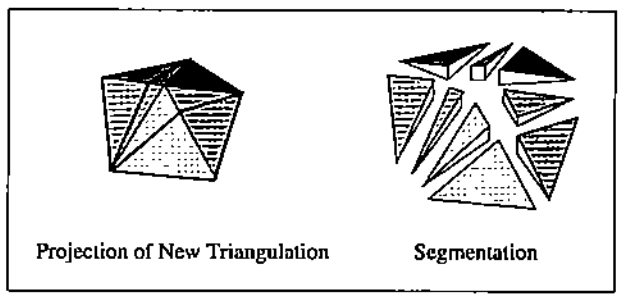



FIGURE 4: Segmentation of mutual projection

### 4.3.1 Geometric Error

Errors in the geometry are quantified by the signed distance spanned by the mapping from one triangulation to another. We use the convention that a displacement toward the "outside" (in the direction of the normal) of a mesh is a positive displacement, while displacement toward the "inside" is a negative displacement. In this way, we are able to compute the introduced geometric error incurred through retriangulation.

## 4.4 Retriangulation

The first step in computing a retriangulation of the neighbors of $v$ is to compute an initial triangulation. In order to accurately capture edge features, we modify this portion of the algorithm to detect edges in the triangulation of $v$ by computing the dihedral angle for each pair of adjacent triangles incident to $v$. Edges are classified as features by a given threshold angle, as in [28]. In the case of two feature angles incident to $v$, our initial triangulation is constrained to contain the edge defining the feature. If three or more feature edges are incident with $v$, the vertex will not be deleted.

Retriangulation proceeds by flipping interior edges which decrease the error introduced by the deletion of $v$. In order to maintain the computed features, edges introduced according to the dihedral angle condition are designated as "unflipable".

With errors computed for each new triangle, we now determine whether the resulting triangulation exceeds the error bound specified by the user. If the triangulation is valid, the candidate vertex and all neighboring triangles are deleted, and the new triangulation is added to the mesh.

## 4.5 Feature Classification and Normals Estimation

We need to *tag* edges of the decimated mesh that correspond to sharp features in the data, and to estimate normals at vertices of the mesh. This information will be used in the data fitting phase.

As we said above, sharp (rectilinear or curvilinear) creases are detected in the initial, fine mesh by looking at the dihedral angle between adjacent faces. Where three ot more edges meet at a vertex we have a sharp corner. When an edge is recognized and tagged as sharp, it is never flipped during the decimation. Points between two adjacent sharp edges can however be removed, as long as the other bounds and conditions of the decimation algorithm are satisfied. Points removed along sharp features are kept in a list to be used later in the data fitting step.

For every vertex in the decimated mesh we need to estimate one (more in the case of a vertex lying on sharp feature) normals. Some of the possible cases are depicted in Figure 5. The simplest case is that of a *smooth* vertex. None of the edges incident on the vertex is tagged as sharp, and the normals of all incident triangles form small pairwise angles. In this case a unique normal (computed by averaging all triangles normals) is associated to the vertex. Notice that even if all edges incident on a vertex are not sharp, the vertex itself can still be singular. This is for example the case of the apex of a cone. This condition is easily checked by looking at the angle formed by normals to pairs of incident faces. In this case, no normals is defined for the vertex, and the vertex is tagged as *singular*. A mix of the previous situations occurs when two or more sharp edges are incident on a vertex. These edges partition the region around the vertex into *sectors*, for each of which we

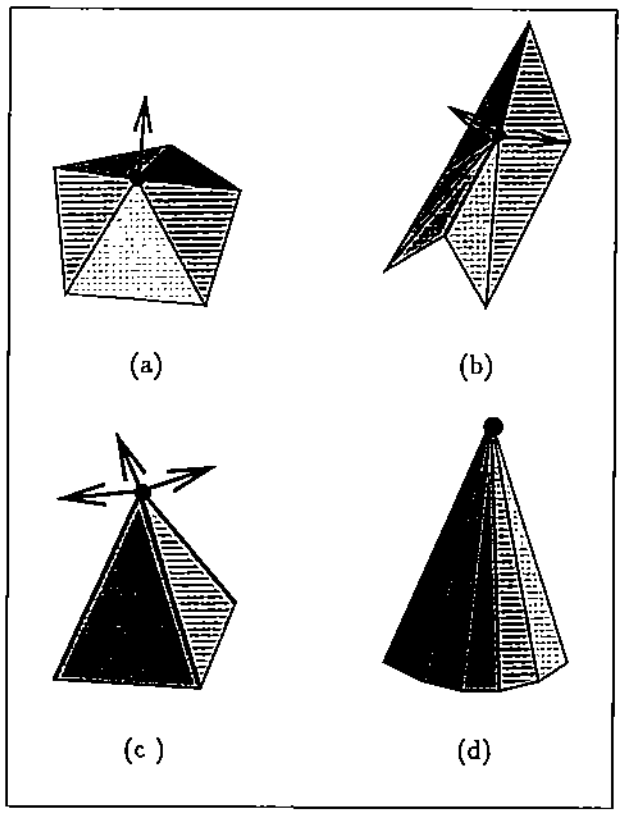



FIGURE 5: Feature classification and normal estimations: A smooth vertex in which a unique (average) normal is defined (a). Two edges along a crease (b); two distinct normals are defined at the vertex. Several sharp edges meeting at a corner (c); each sector between two sharp edges has its own normal. A singular vertex (d); no normal is defined.

will compute a separate normals. Again, the group of triangles forming a sector can have similar normals (therefore defining a smooth sector with a unique, averaged normal) or rather divergent, in which case the sector is singular and no normal is defined for it.

In summary, we can say that the region around a vertex can be divided into one or more sectors, and each sector has an associated normals (undefined in the case of a singular sector). Notice that normals can be computed from the initial fine mesh, and can therefore provide a rather good estimate of the surface curvature.

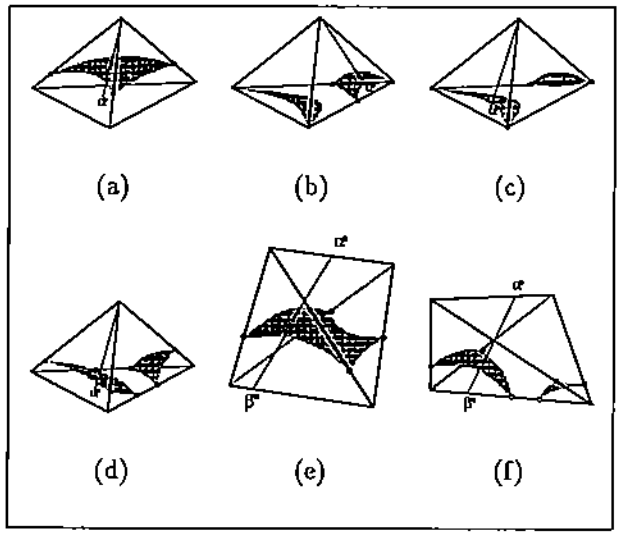



FIGURE 6: Three-sided ((a), (b), (c) and (d)) and four-sided patches ((e) and (f)).

# 5 Piecewise-Smooth Reconstruction

In the final step of our algorithm we fit the data points with polynomial patches. For each triangle of the decimated mesh we compute a patch of surface that interpolates its vertices and estimated vertex normals if required by the continuity conditions.

We use a particular type of algebraic Bernstein-Bézier patches of degree three, called *A-patches* [3], for our fitting. Here we shortly review definitions and properties of A-patches. A more detailed discussion can be found in the cited paper.

## 5.1 A-patches

A cubic Bernstein-Bézier polynomial over a tetrahedron can be written as [13]

$$f(p) = \sum_{\lambda_1+\lambda_2+\lambda_3=3} b_\lambda \frac{3!}{\lambda_1!\lambda_2!\lambda_3!} \tau_1^{\lambda_1}\tau_2^{\lambda_2}\tau_3^{\lambda_3}$$

where $\tau$ are the barycentric coordinates of $p$ w.r.t. the tetrahedron, and $b_\lambda$ are the *Bézier* ordinates (or weights. A cubic Bernstein-Bézier polynomial has 20 weights). The regular lattice of Bézier points $(\lambda/3, b_\lambda)$ is called Bézier net and many properties of these polynomials (E.g., continuity between two adjacent patches) can be expressed in terms of geometric conditions on this net.

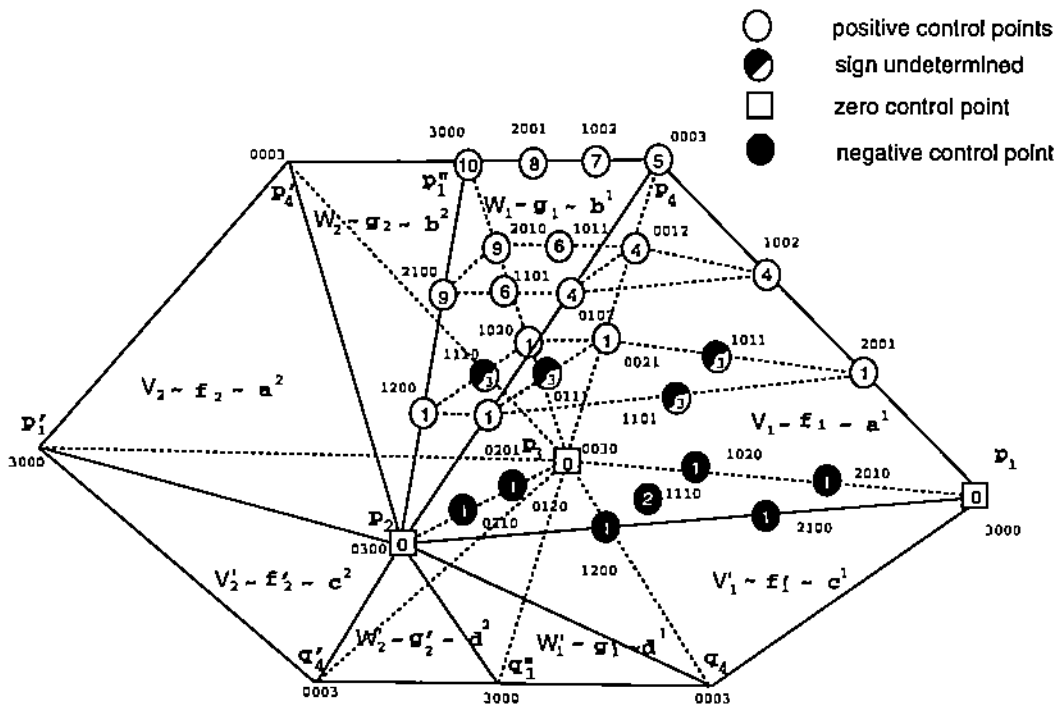



FIGURE 7: Weights for a $C^1$ patch.

For example, two patches are $C^0$-continuous across a common face (or edge, or vertex) if and only if each weight of one patch on that face is equal to the corresponding weight of the other patch.

The following lemma gives necessary and sufficient conditions for $C^0$ and $C^1$ continuity:

**Lemma 5.1** *(see [14]). Let* $f(p) = \sum_{|\lambda|=n} a_\lambda B^n_\lambda(\alpha)$ *and* $g(p) = \sum_{|\lambda|=n} b_\lambda B^n_\lambda(\alpha)$ *be two polynomials defined on two tetrahedra* $[p_1p_2p_3p_4]$ *and* $[p'_1p_2p_3p_4]$*, respectively. Then*

*(i)* $f$ *and* $g$ *are* $C^0$ *continuous at the common face* $[p_2p_3p_4]$ *if and only if*

$$a_\lambda = b_\lambda, \quad \text{for all } \lambda = 0\lambda_2\lambda_3\lambda_4, \quad |\lambda| = n \quad (1)$$

*(ii)* $f$ *and* $g$ *are* $C^1$ *continuous at the common face* $[p_2p_3p_4]$ *if and only if (1) holds and, for all* $\lambda = 0\lambda_2\lambda_3\lambda_4$, $|\lambda| = n-1$,

$$b_{\lambda+e_1} = \beta_1 a_{\lambda+e_1} + \beta_2 a_{\lambda+e_2} + \beta_3 a_{\lambda+e_3} + \beta_4 a_{\lambda+e_4} \quad (2)$$

*where* $\beta = (\beta_1, \beta_2, \beta_3, \beta_4)^T$ *is defined by the following relation*

$$p'_1 = \beta_1 p_1 + \beta_2 p_2 + \beta_3 p_3 + \beta_4 p_4, \quad |\beta| = 1$$

When relation (2) is satisfied, we say that the weights involved are *coplanar*.

A piecewise algebraic surface patch [29, 30] is the region of the zero-contour of a Bernstein-Bézier polynomial contained within the tetrahedron, and an A-patch is an algebraic surface patch satisfying certain smoothness conditions. In particular, A-patches are guaranteed, by imposing constraints on the values of their weights, to be non singular (except where needed, see below) and single-sheeted. In particular, A-patches can be classified as *three-sided* when a segment connecting a vertex of the tetrahedron to a points on the opposite face intersects the patch at most once, and *four-sided* when the property holds for a segment connecting two points on two opposite edges (see Figure 6).

We have just seen that an A-patch is defined inside a tetrahedron. We therefore need to build these tetrahedra from the triangle mesh obtained in the previous step. Notice that it is not sufficient to associate a tetrahedron to each triangle of the mesh, as we need to fill the gaps that would be formed along the edges of the mesh. We therefore build two types of tetrahedra: *face*-tetrahedra, associated with triangles of the mesh, and *edge*-tetrahedra bridging two face-tetrahedra which share that edge. The tetrahedron

mesh obtained with this process is called *simplicial hull* (see Figure 1(d) for an example), and details on its construction can be found in [3].

### 5.2 Data fitting

Once the simplicial hull is constructed, we need to set the weights of each patch so that the surface is $C^1$ (or $C^0$ in the presence of a sharp feature) and the collection of patches approximate the data points.

$C^0$ and $C^1$ features can be mixed into the same model, by appropriately setting weight and allowing patches with singular vertices/edges (weights around the vertices/edges are all zero, compared to coincident control points in the parametric case). Figure 8 shows how $C^0$ and $C^1$ features can be mixed to approximate a cube in different shapes.

Similar to the trivariate interpolation case, Powell-Sabin or Clough-Tocher splits are used to introduce degree-bounded vertices to prevent the continuity system from propagating globally. Such splitting, however, could result in a large number of patches. A full trivariate Clough-Tocher split would split a tetrahedron into 12 sub-tetrahedra. However, as only the zero set of the polynomial is of interest, one does not need a complete mesh covering the entire space. Furthermore, a vertex does not need to be fully covered by the trihedral corners of the incident tetrahedra. A "incomplete" vertex helps introduce degree-bounded vertices as well. An effective simplicial hull construction of this kind first appears in Dahmen [9], and subsequently developed and used by Guo [17], Lodha [24], Dahmen and Thamm-Scharr [10], Bajaj, Chen and Xu [3].

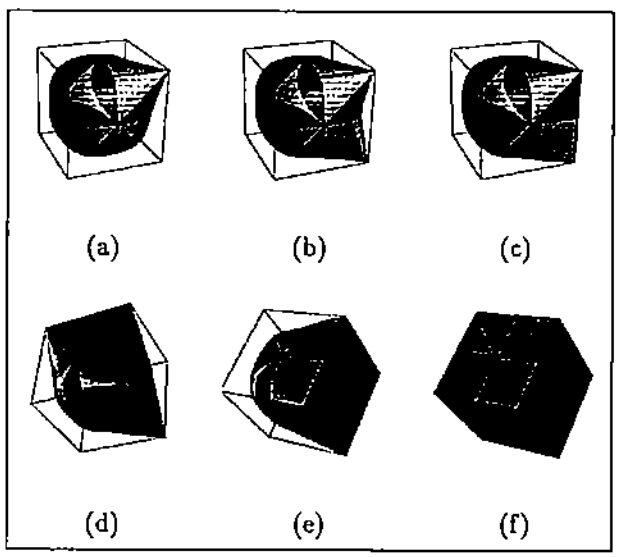



FIGURE 8: Modeling with singular A-patches. (a) Interpolating a vertex with a singular point. (b) Interpolating two vertices. (c) Interpolating an edge with a singular edge on the surface. (d) Interpolating two edges. (e) Interpolating a face of a cube. (f) The A-patch surface degenerates into the cube. All the edges are now singular.

Figure 7 shows the scheme to set the weights of a patch under $C^1$ continuity constraints (the patch interpolates vertices and vertex normal of its base triangle). The sequence of steps involved is as follows:

1. Set weights ⓪ 's to be zero so that the surface interpolates the vertices. Set weights ① 's around each vertex according to its normal so that the surface interpolates the normal. Set weights ② 's according to some optimal criteria.

2. Weights ① 's, ② 's and ③ 's around $[p_2p_3]$ must be affine coplanar. Solve for ③ 's according to the others.

3. Compute weights ⑥ 's according to weights ④ 's, ③ 's and ① 's. Preset weights ④ 's to be positive enough so that weights ⑥ 's are also positive.

4. Compute weights ⑦ 's according to weights ⑤ 's and ④ 's. Preset weights ⑤ 's to be positive enough so that weights ⑦ 's are also positive.

5. Set weights ⑧ 's to be positive. Compute weights ⑨ 's by averaging ⑥ 's and ⑧ 's.

## 6 Results and Conclusions

Examples of reconstruction of 3D models are presented and discussed in this Section.

Figure 9 contains a dense sampling of points from a high-heeled shoe. The original sampling consists of $2 \cdot 10^4$ points, uniformly distributed over the surface. The reconstructed piecewise linear object contains 17786 triangles. This linear approximation is decimated to a rough approximation of 248 triangles, while maintaining $C^0$ features and a surface distance error of less than 1%. A total of about 1500 piecewise $C^0$ and $C^1$ cubic patches interpolate the decimated model and approximate the original set of points. The entire process required approximately 2 minutes

for the computation of the $\alpha$-solid and successive decimation, less that a minute to set the weights of the polynomial patches in order to satisfy the continuity conditions, and about 20 minutes to improve the error of fit by least squares approximation of the original points. All computations were carried out on an SGI Indigo2.

## References


[1] Bajaj, C., Bernardini, F., and Xu, G. Automatic reconstruction of surfaces and scalar fields from 3D scans. In *Computer Graphics Proceedings* (1995), Annual Conference Series. Proceedings of SIGGRAPH 95, ACM SIGGRAPH, pp. 109–118.

[2] Bajaj, C., Chen, J., and Xu, G. Free form surface design with A-patches. In *Proceedings of Graphics Interface '94* (Banff, Canada, 1994), pp. 174–181.

[3] Bajaj, C., Chen, J., and Xu, G. Modeling with cubic A-patches. *ACM Transactions on Graphics 14*, 2 (1995), 103–133.

[4] Bajaj, C., and Schikore, D. Error-bounded reduction of triangle meshes with multivariate data, to appear. In *Proceedings of SPIE Symposium of Visual Data Exploration and Analysis* (Jan. 1996), G. Grinstein and R. Erbacher, Eds.

[5] Barnhill, R. E. Surfaces in computer aided geometric design: A survey with new results. *Computer Aided Geometric Design 2* (1985), 1–17.

[6] Bolle, R. M., and Vemuri, B. C. On three-dimensional surface reconstruction methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 1 (Jan. 1991), 1–13.

[7] Chen, X., and Schmitt, F. Surface modelling of range data by constrained triangulation. *Computer Aided Design 26*, 8 (Aug. 1994), 632–645.

[8] Choi, B. K., Shin, H. Y., Yoon, Y. I., and Lee, J. W. Triangulation of scattered data in 3D space. *Computer Aided Design 20*, 5 (June 1988), 239–248.

[9] Dahmen, W. Smooth piecewise quadratic surfaces. In *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. Schumaker, Eds. Academic Press, Boston, 1989, pp. 181–193.

[10] Dahmen, W., and Thamm-Schaar, T.-M. Cubicoids: modeling and visualization. *Computer Aided Geometric Design 10* (1993), 93–108.

[11] Edelsbrunner, H. Weighted alpha shapes. Tech. Rep. UIUCDCS-R-92-1760, Comput. Sci. Dept., Univ. Illinois, Urbana, Ill, 1992.

[12] Edelsbrunner, H., and Mücke, E. P. Three-dimensional alpha shapes. *ACM Trans. Graph. 13*, 1 (Jan. 1994), 43–72.

[13] Farin, G. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design 3* (1986), 83–127.

[14] Farin, G. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* Academic Press, 1990.

[15] Faugeras, O. D., Hebert, M., Mussi, P., and Boissonnat, J. D. Polyhedral approximation of 3-D objects without holes. *Computer Vision, Graphics and Image Processing 25* (1984), 169–183.

[16] Goshtasby, A. Design and recovery of 2-D and 3-D shapes using rational Gaussian curves and surfaces. *International Journal of Computer Vision 10*, 3 (1993), 233–256.

[17] Guo, B. *Modeling arbitrary smooth objects with algebraic surfaces.* PhD thesis, Computer Science, Cornell University, 1991.

[18] Hamann, B. A data reduction scheme for triangulated surfaces. In *Computer Aided Geometric Design* (1994), vol. 13, pp. 197–214.

[19] He, T., Hong, L., Kaufman, A., Varshney, A., and Wang, S. Voxel based object simplification. In *Visualization '95 Proceedings* (Oct. 1995), G. M. Nielson and D. Silver, Eds., pp. 296–303.

[20] Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schwitzer, J., and Stuelzle, W. Piecewise smooth surface reconstruction. In *Computer Graphics Proceedings* (1994), Annual Conference Series. Proceedings of SIGGRAPH 94, ACM SIGGRAPH, pp. 295–302.

[21] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuelzle, W. Surface reconstruction from unorganized points. *Computer Graphics 26*, 2 (July 1992), 71–78. Proceedings of SIGGRAPH 92.

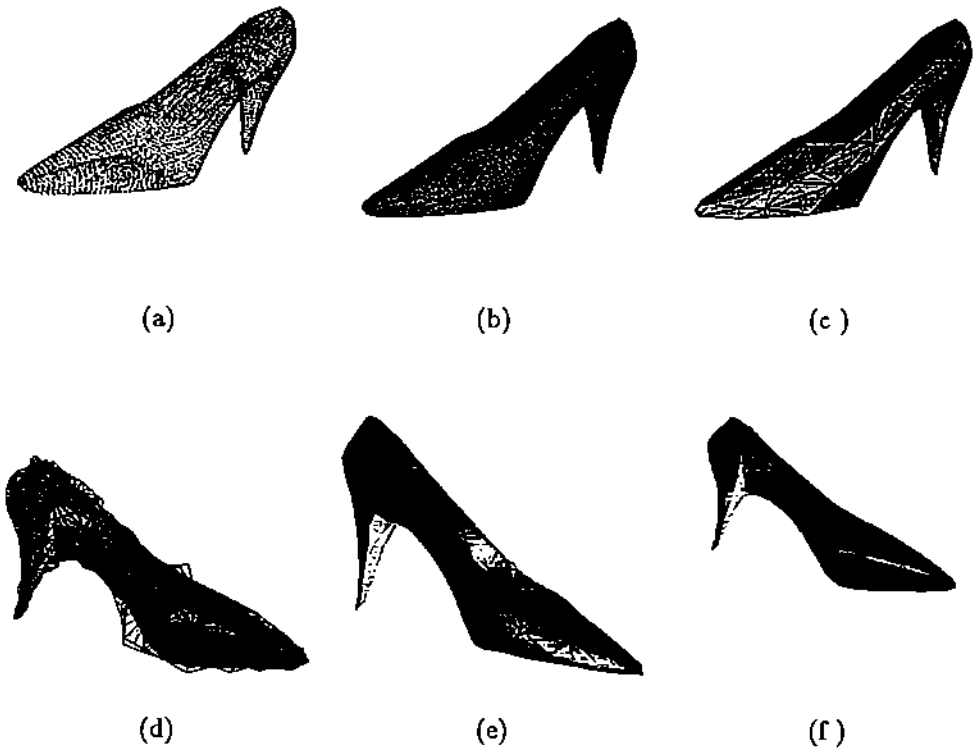



FIGURE 9: Example of an object reconstruction. The shoe was sampled at about $2 \cdot 10^4$ points (a). The selected $\alpha$-solid is shown in (b). The decimated mesh, 310 triangles (c) is used in (d) to build the simplicial hull and fit the data with polynomial algebraic patches (e). The final result is shown in (f).

[22] HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. Mesh optimization. In *Computer Graphics (SIGGRAPH '93 Proceedings)* (Aug. 1993), J. T. Kajiya, Ed., vol. 27, pp. 19–26.

[23] LIAO, C. W., AND MEDIONI, G. Surface approximation of a cloud of 3D points. *Graphical Models and Image Processing 57*, 1 (Jan. 1995), 67–74.

[24] LODHA, S. *Surface Approximation with Low Degree Patches with Multiple Representations.* PhD thesis, Computer Science, Rice University, Houston, Texas, 1992.

[25] MOORE, D., AND WARREN, J. Approximation of dense scattered data using algebraic surfaces. In *Proceedings of the 24th annual Hawaii International Conference on System Sciences* (1991), V. Milutinovic and B. D. Shriver, Eds., vol. 1.

[26] PATRIKALAKIS, N. M., AND KRIEZIS, G. A. Representation of piecewise continuous algebraic surfaces in terms of B-splines. *The Visual Computer 5* (1989), 360–374.

[27] SCHMITT, F., BARSKY, B. A., AND DU, W. An adaptive subdivision method for surface fitting from sampled data. *Computer Graphics 20*, 4 (1986), 179–188. Proceedings of SIGGRAPH 86.

[28] SCHROEDER, W. J., ZARGE, J. A., AND LORENSEN, W. E. Decimation of triangle meshes. In *Computer Graphics (SIGGRAPH '92 Proceedings)* (July 1992), E. E. Catmull, Ed., vol. 26(2), pp. 65–70.

[29] SEDERBERG, T. Techniques for cubic algebraic surfaces, part one. *IEEE Computer Graphics & Applications 10*, 4 (July 1990), 14–25.

[30] SEDERBERG, T. Techniques for cubic algebraic surfaces, part two. *IEEE Computer Graphics & Applications 10*, 6 (Sept. 1990), 12–21.

[31] SEDERBERG, T. W. Piecewise algebraic surface patches. *Computer Aided Geometric Design 2* (1985), 53–59.

[32] TURK, G. Re-tiling polygonal surfaces. In *Computer Graphics (SIGGRAPH '92 Proceedings)* (July 1992), E. E. Catmull, Ed., vol. 26(2), pp. 55–64.

[33] VELTKAMP, R. C. *Closed object boundaries from scattered points.* PhD thesis, Center for Mathematics and Computer Science, Amsterdam, 1992.